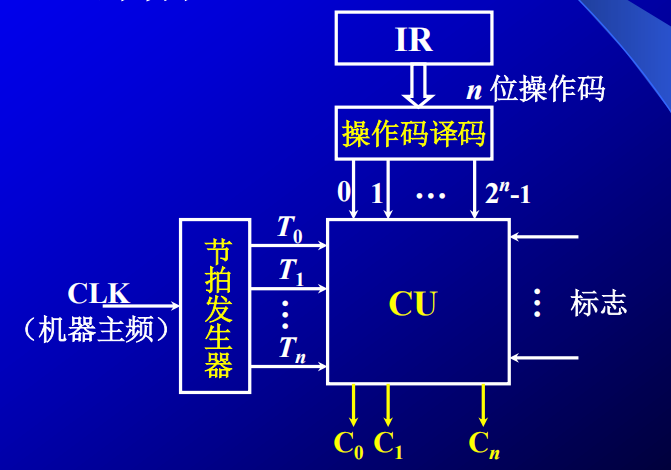
**10.1 组合逻辑设计**

**一、组合逻辑控制单元框图**



节拍信号Ti：循环序列，在第i个周期时拉高

**二、微操作的节拍安排**

采用 同步控制方式

一个 机器周期 内有 3个节拍（时钟周期）

CPU 内部结构采用非总线方式

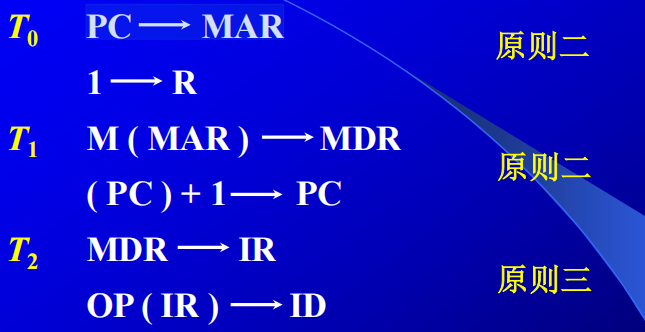
**安排微操作时序**的原则：

原则一：微操作的**先后顺序**不得随意更改

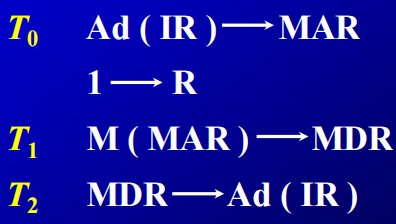
原则二：**被控对象不同**的微操作，尽量安排在**一个节拍内**完成（尽量并行）

原则三：**占用时间较短**的微操作，尽量安排在**一个节拍内**完， 并**允许有先后顺序**

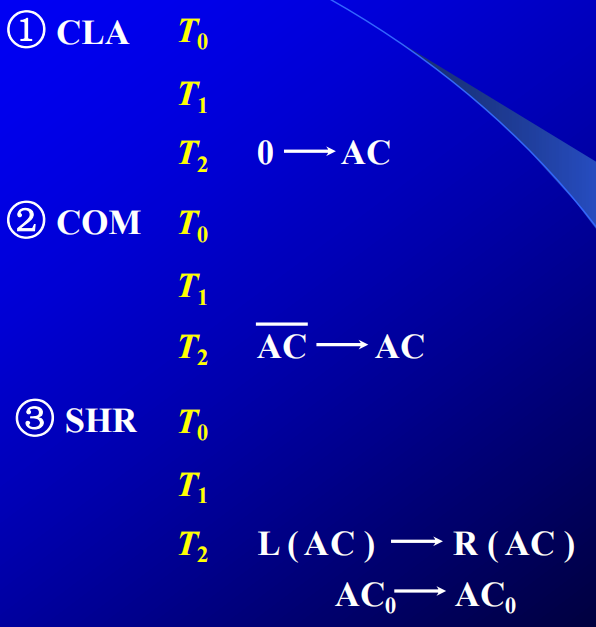
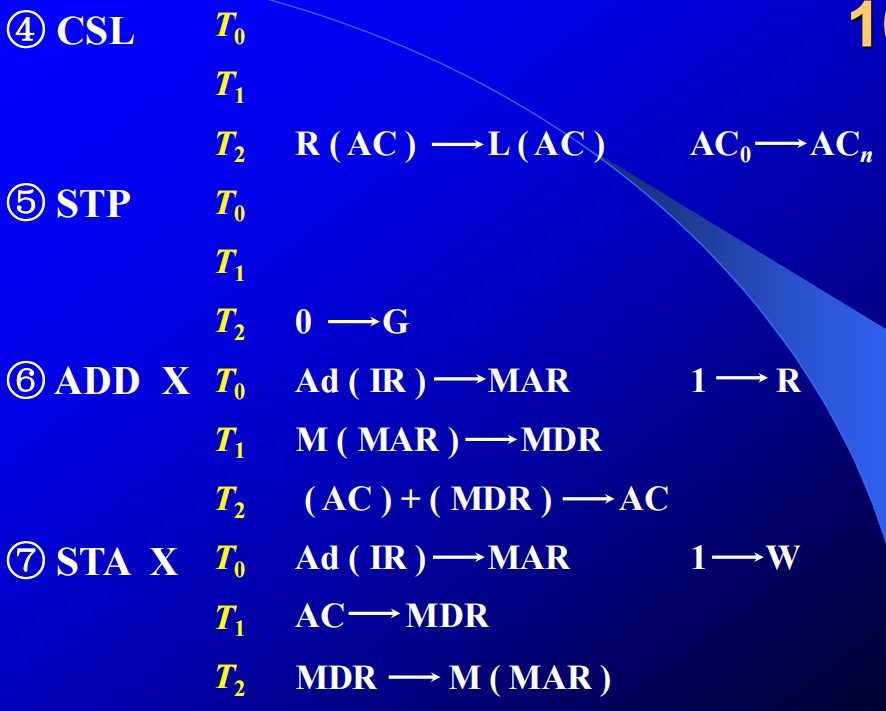
**取指周期微操作的节拍安排**

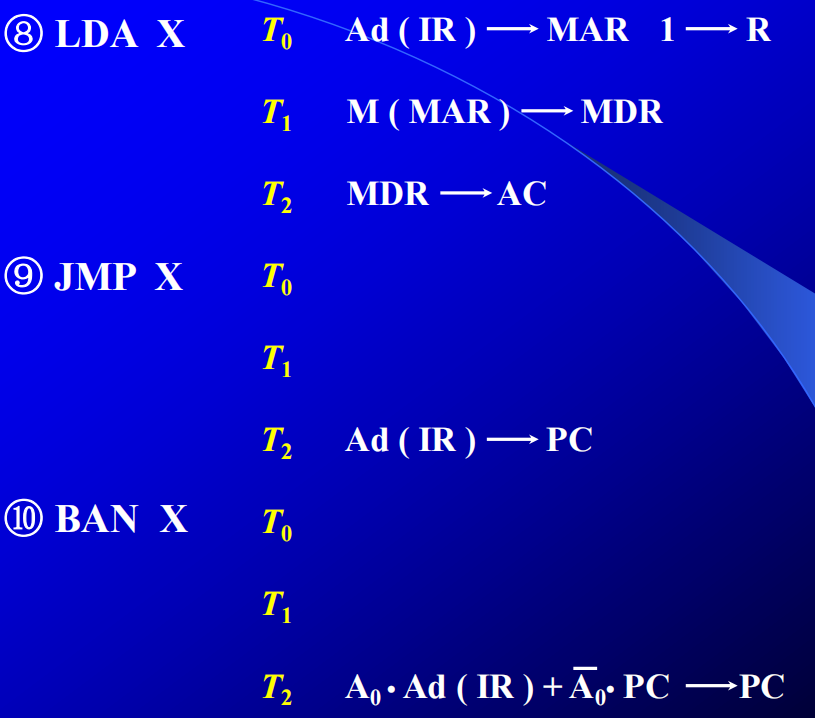


**间址周期微操作的节拍安排**



**执行周期微操作的节拍安排**





**中断周期微操作的节拍安排**



**三、组合逻辑设计步骤**

1. 列出操作时间表

2. 写出微操作命令的最简表达式

3. 画出逻辑图

**10.2 微程序设计**

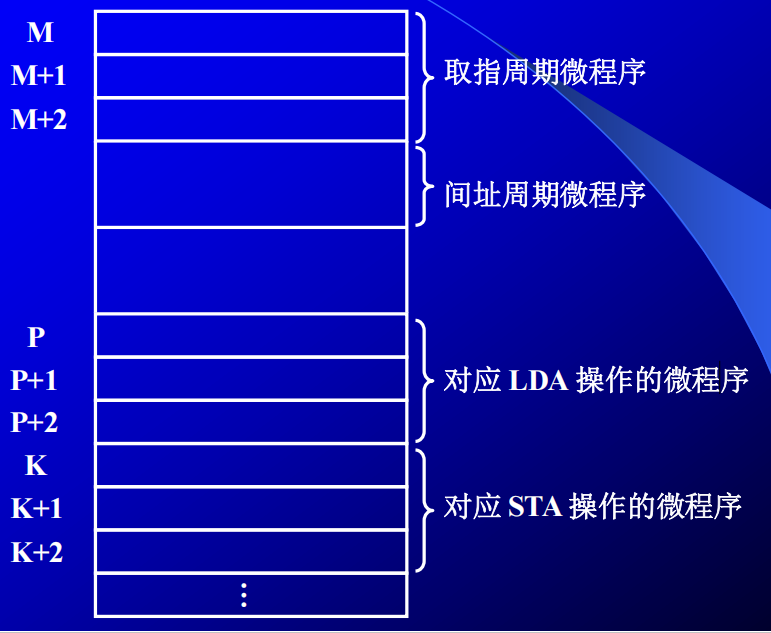
**一、微程序设计思想的产生**

一条机器指令对应一个微程序 存入ROM

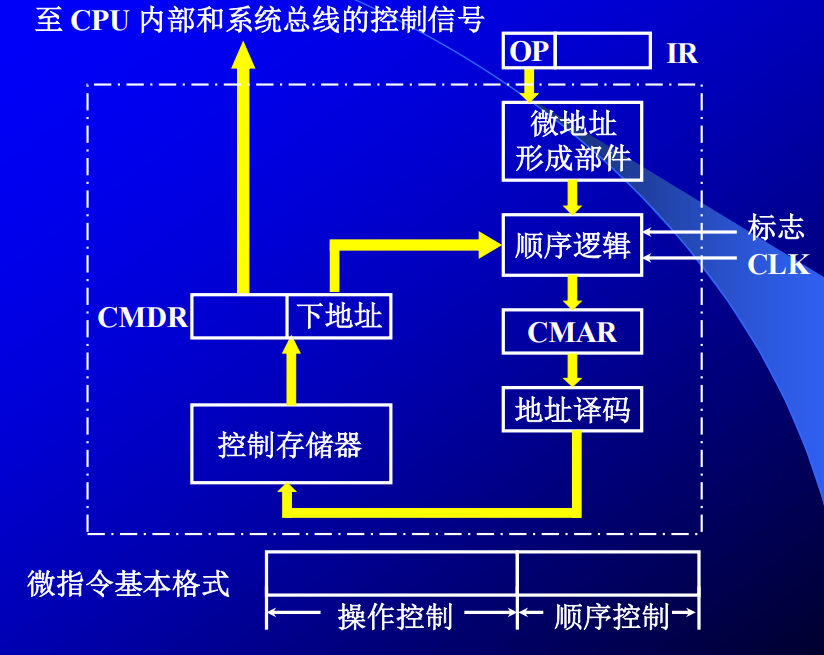
完成一条机器指令/一个微程序，需要完成多个微操作命令

**二、微程序控制单元框图及工作原理**

**1. 机器指令对应的微程序**



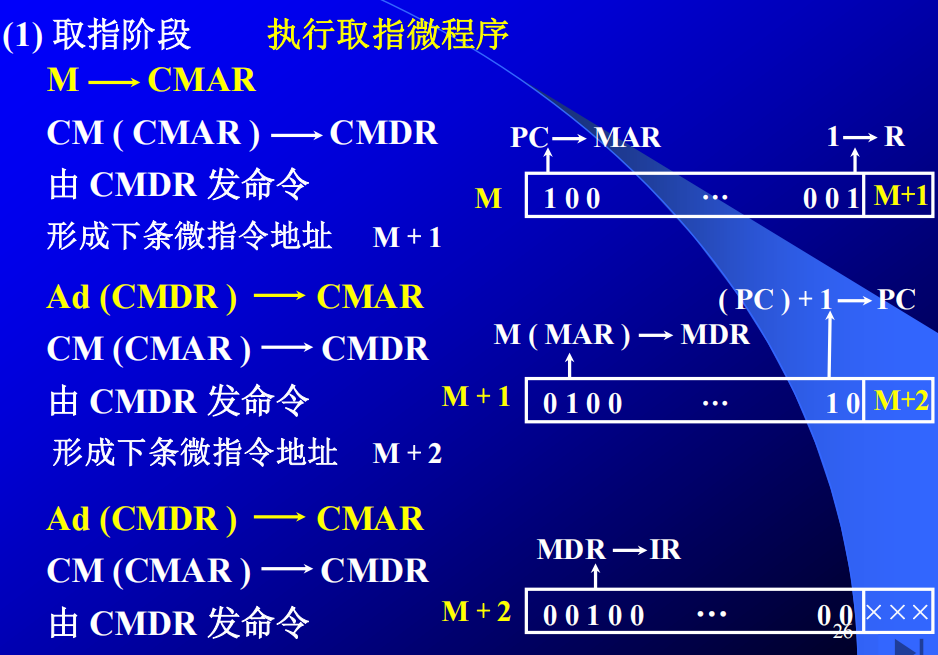
**2. 微程序控制单元的基本框图**

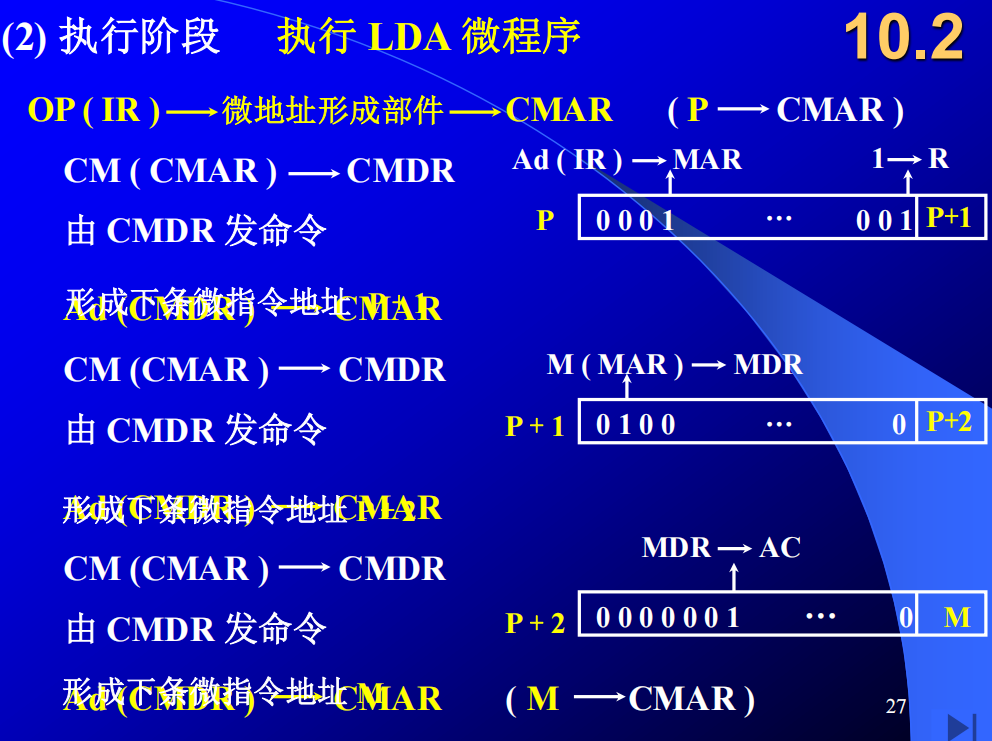


**3. 工作原理**

**关键： ➢ 微指令的 操作控制字段如何形成微操作命令**

**➢ 微指令的 后续地址如何形成**







**三、微指令的编码方式（控制方式）**

1. 直接编码（直接控制、不译码）方式

在微指令的操作控制字段中，**每一位代表一个微操作命令，**‘1’表示该信号有效

**译码速度最快，但占用字长较长**

2. 字段直接编码方式

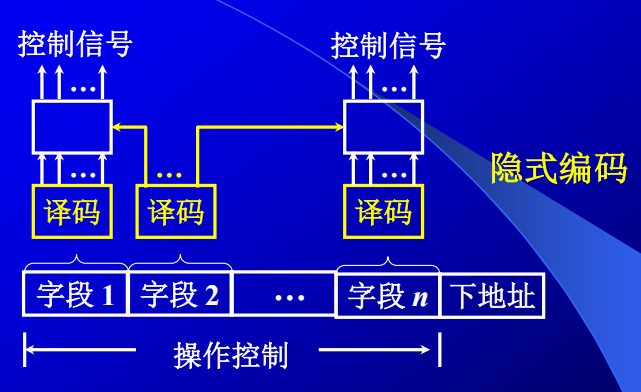
将微指令的控制字段分成若干“段”，每段经译码后发出控制信号

（显式编码：一个控制信号由一个字段译码决定）

**每个字段中的命令是互斥的**

**缩短了微指令字长，但增加了译码时间**

3. 字段间接编码方式



隐式编码：一个控制信号需要经过多个字段译码决定

4. 混合编码

直接编码和字段编码（直接和间接）混合使用

5. 其他

微指令中设常数字段等

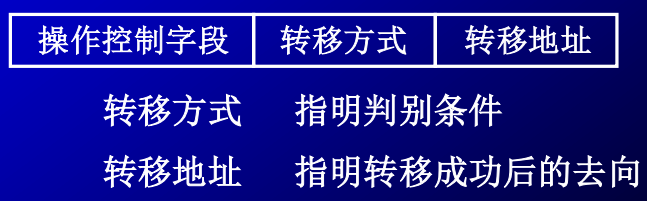
**四、微指令序列地址的形成**

1. 微指令的 **下地址字段** 指出

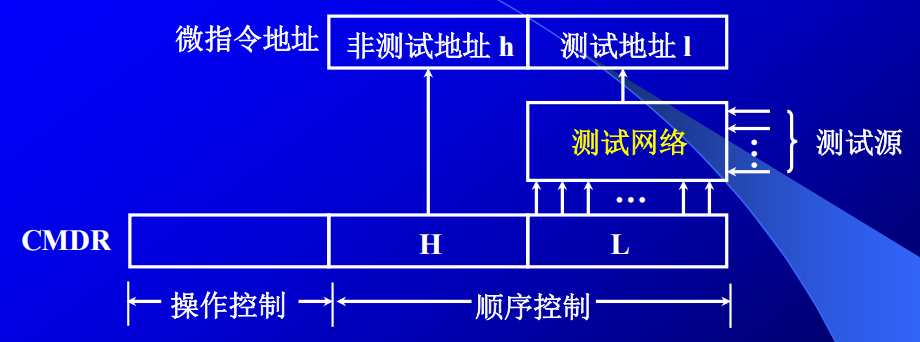
2. 根据机器指令的 操作码 形成

3. 增量计数器：( CMAR ) + 1 -> CMAR

4. 分支转移



5. 通过测试网络

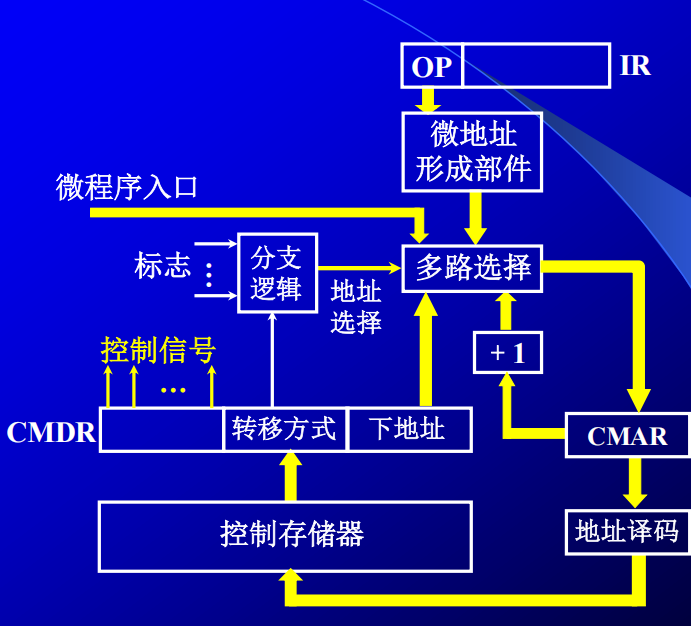


6. 由硬件产生微程序入口地址

第一条微指令地址：由专门硬件产生

中断周期：由硬件产生**中断周期微程序首地址**

7. 后续微指令地址形成方式原理图



**五、微指令格式**

1. 水平型微指令

一次能定义并执行多个**并行**操作

如 直接编码、字段直接编码、字段间接编码、直接和字段混合编码

2. 垂直型微指令

类似机器指令操作码的方式，并行能力不强，但有严格的顺序结构

由微操作码字段规定微指令的功能

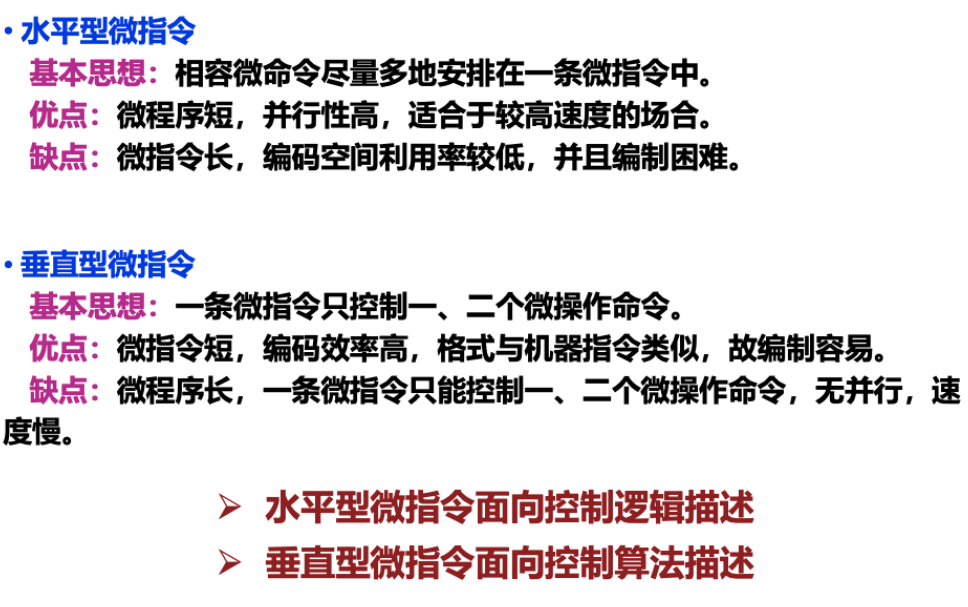
3. 两种微指令格式的比较

(1) 水平型微指令比垂直型微指令 并行操作能力强，灵活性强

(2) 水平型微指令执行一条机器指令所要的微指令数目少，速度快

(3) 水平型微指令 用**较短**的**微程序**结构换取**较长**的**微指令**结构

(4) 水平型微指令与机器指令差别大



**六、静态微程序设计和动态微程序设计**

静态 微程序无须改变，采用 ROM

动态：改变微指令和微程序来改变机器指令，有利于仿真，采用EPROM(可擦只读存储器)

**七、毫微程序设计**

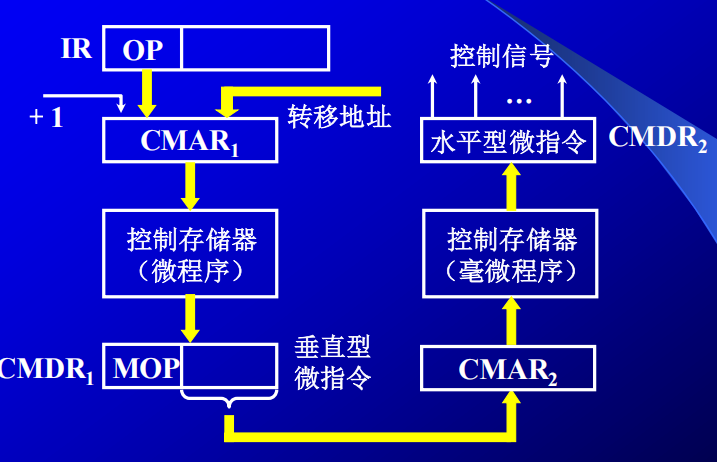
1. 毫微程序设计的基本概念

微程序设计 用 微程序解释机器指令

毫微程序设计 用 毫微程序解释微指令

毫微指令与微指令 的关系好比 微指令与机器指令 的关系

2. 毫微程序控制存储器的基本组成



**九、微程序设计举例**

具体步骤：首先写出对应机器指令的全部微操作及节拍安排，然后确定微指令格式，最后编写出每条微指令的二进制代码（称为微指令码点）

